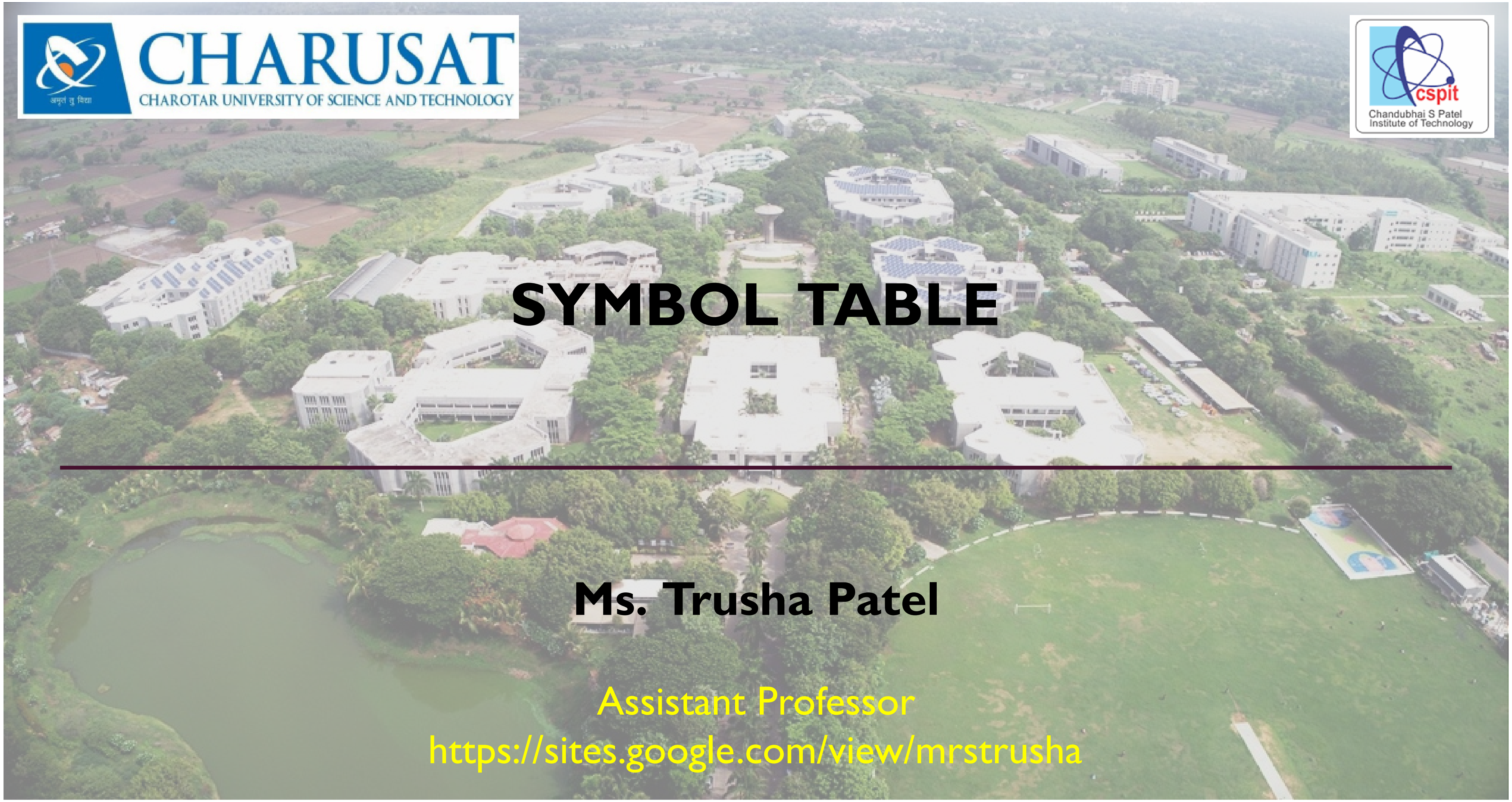

SYMBOL TABLE

Ms. Trusha Patel

Assistant Professor
https://sites.google.com/view/mrstrusha

WHAT IS SYMBOL TABLE?

- A compiler uses symbol table to keep track of scope and binding information about names

- Operation on symbol table

  - Search : Search every time a new name is encounter
  - Add : Add if new name encounter
  - Update : Changes occur when new information of existing name is encounter

- Symbol table size must be dynamic for it can grow (if necessary) at run time

SYMBOL TABLE ENTRIES

- Format of entries does not have to be uniformed
  because the information saved about the name depend on the usage of the name

Example :

Class of Name	Information
Variable	type, length, dimension information
Procedure name	address of parameter list, number of parameters
Function name	Type of returned value, length of returned value, address of parameter list, number of parameters
label	Statement number

SYMBOL TABLE ENTRIES

- To keep symbol-table records uniform, it may be convenient for some of the information about a name to be kept outside the table entry, with only a pointer to this information stored in the record

- Information is entered in table at various time
  - Keywords (if any) are entered initially
  - Lexical analyser make entry of identifiers (variable, function name etc.)

SYMBOL TABLE ENTRIES

- If there is a modest upper bound on the length of a name, then the characters in the name can be stored in the symbol-table entry, as in Fig.(1)

Name										Attribute
s	o	r	t							
a										
s	t	u	_	n	a	m	e			
x	y	z								

Fig.(1) Fix length entry

SYMBOL TABLE ENTRIES

- If there is no limit on the length of a name, or if the limit is rarely reached, the indirect scheme of Fig.(2) can be used.

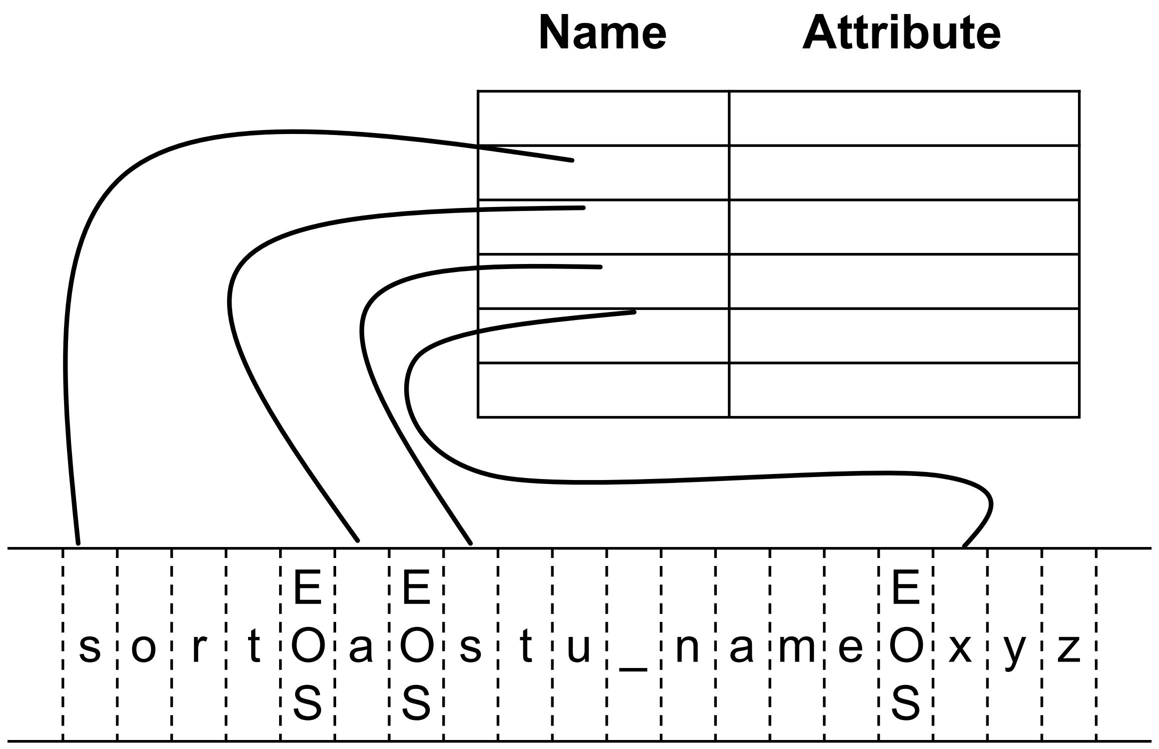

Fig.(2) variable length entry

DATA STRUCTURE FOR SYMBOL TABLE

1. Linear list
2. Hash tables

DATA STRUCTURE FOR SYMBOL TABLE

1. Linear list

- Simplest and easiest o implement symbol table Fig.(3)
- Use single array or equivalent several array to store name and attributes
- End or array marked by pointer "available"

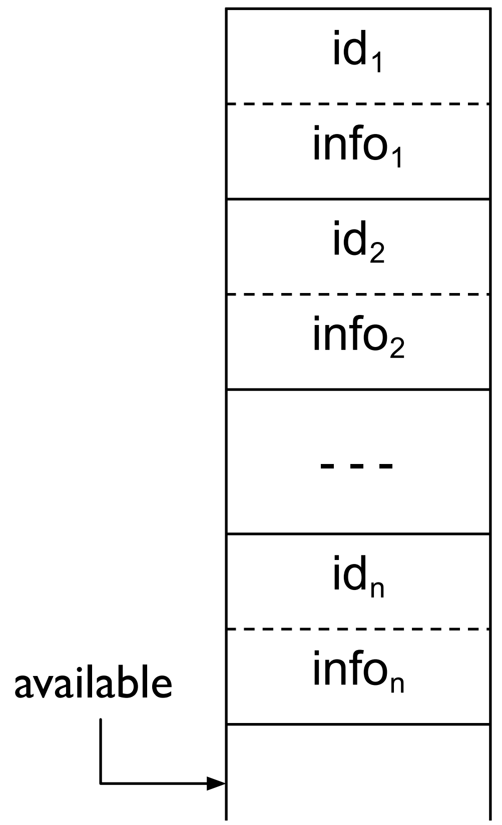

Fig.(3) linear list of records

DATA STRUCTURE FOR SYMBOL TABLE

1. Linear list

- ADD operation
  - New entry is made in space immediately followed by “available”, increase pointer by size of new record
  - New name will be added to the list in the order in which they are encountered
- SEARCH operation
  - Search of a name proceeds from the end of the array (from available) to the beginning
  - When name is located associated information can be found
  - If reach to beginning of list without finding the name then fault occurs

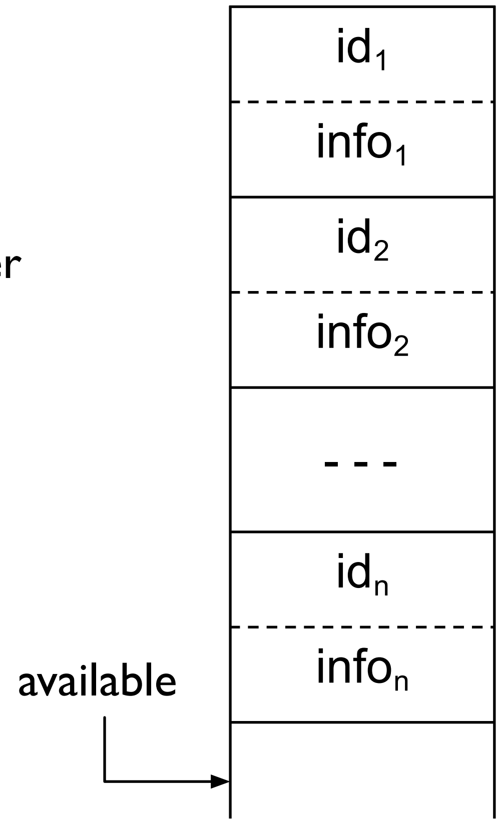

Fig.(3) linear list of records

DATA STRUCTURE FOR SYMBOL TABLE

1. Linear list

- Disadvantage
  - If multiple entries of same name is not allowed then need to look through table before making new entry if "n" total names and "e" inquiries then cost to make "n" entries will be cn(n+e) if "n" and "e" are very high then cost will be too much

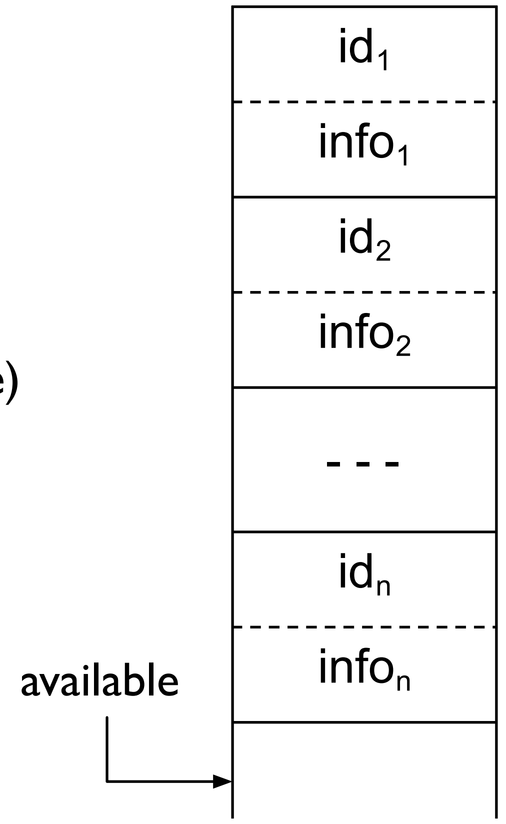

Fig.(3) linear list of records

DATA STRUCTURE FOR SYMBOL TABLE

2. Hash tables

- More efficient than linear list
- Used in many compilers
- Basic hashing scheme shown in Fig.(4)
- There are 2 parts
  - Hash table consist of pointer to able entries
  - Table entries organized in m separate link list (buckets)

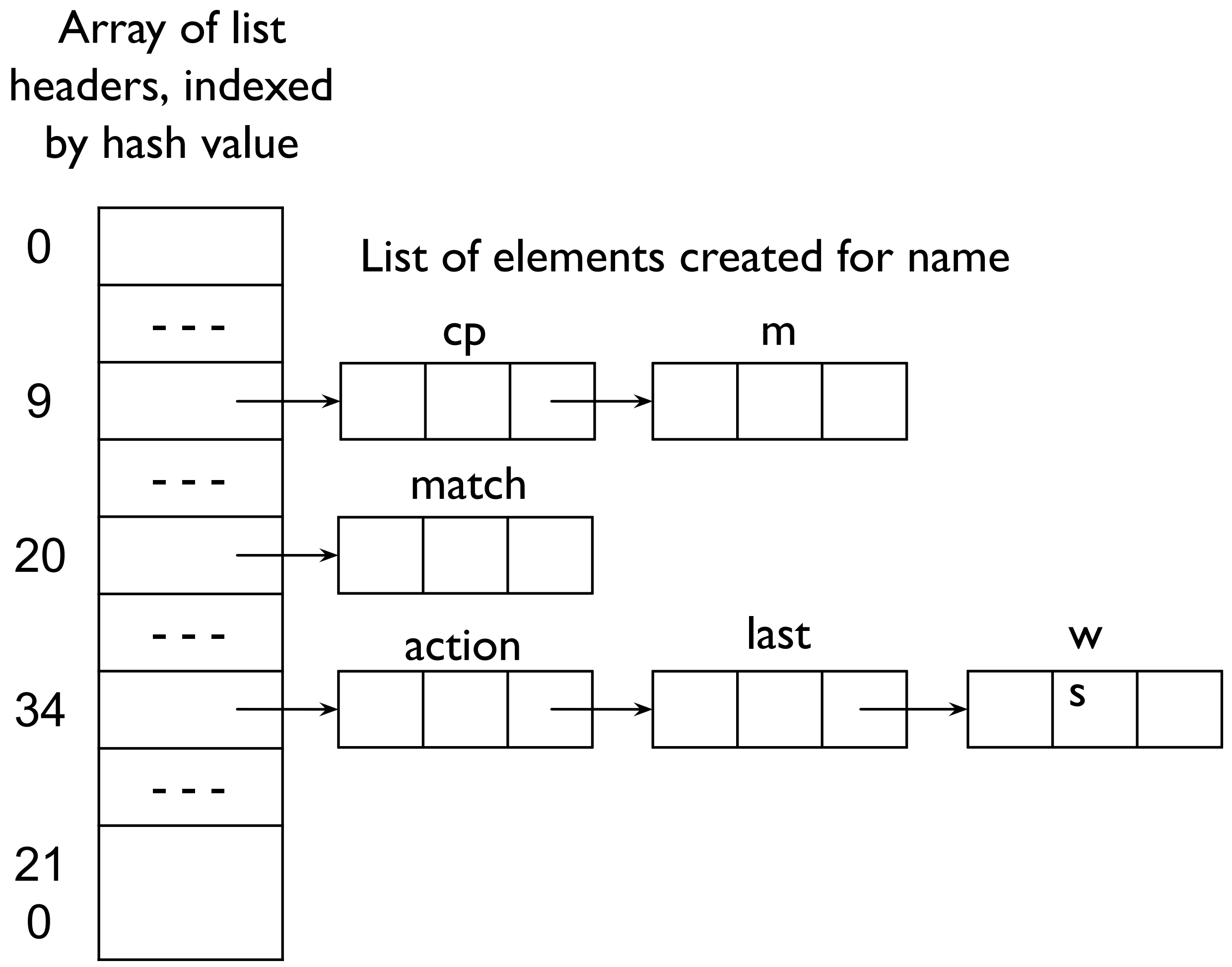
Array of list headers, indexed by hash value
0
- - -
9
- - -
20
- - -
34
- - -
210
List of elements created for name
cp
m
match
action
last
w
s

Fig.(4) hash table of size 211

DATA STRUCTURE FOR SYMBOL TABLE

2. Hash tables

- To determine entry of "s" in symbol table apply hash function "h" on "s" such that h(s) will return integer value between 0 to m-1 (m is hash table size), then it is on the list numbered by h(s)
- If "s" is not in list then enter by creating record of that, linked at front of list numbered by h(s)

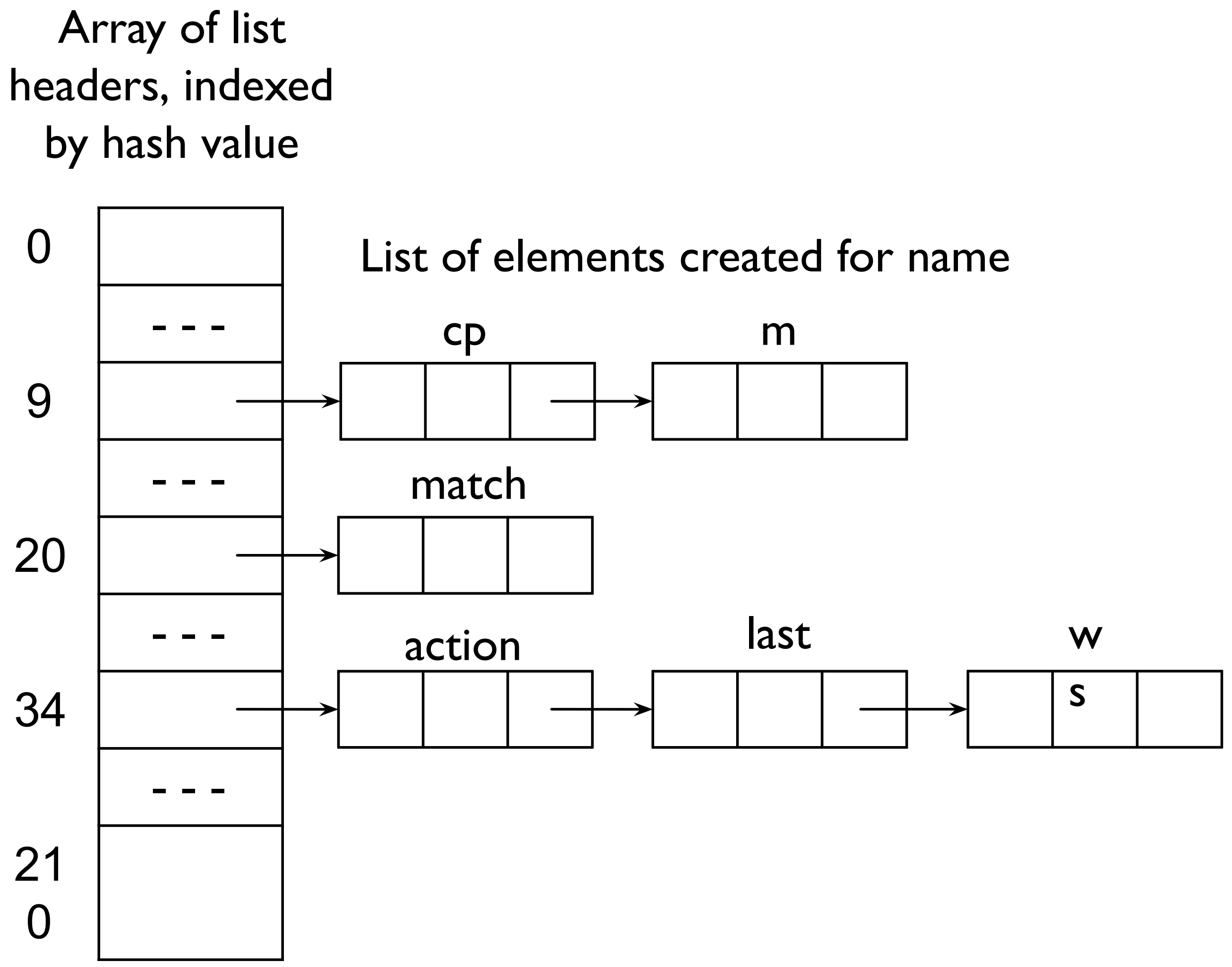
Array of list headers, indexed by hash value
List of elements created for name
0
- - -
9
- - -
20
- - -
34
- - -
210
cp
m
match
action
last
w
s

Fig.(4) hash table of size 211

REPRESENTING SCOPE INFORMATION

- Symbol table maintain information of identifiers
- Scope of all identifiers may be different
- Simple approach to maintain different scope is to maintain a separate symbol table for each scope

REFERENCE

- Alfred Aho, Ravi Sethi, Jeffrey D Ullman, *Compilers Principles, Techniques and Tools*, Pearson Education Asia.